# GNU Emacs Reference Card

(for version 30)

## Key Binding Notation

In the Emacs key binding notation, `C-x` is `Ctrl+X`; `M-x` is usually `Alt+X`; `S-x` is `Shift+X`; and `C-M-x` is `Ctrl+Alt+X`, etc.

## Leaving Emacs

| | |
|---|---|
| iconify Emacs (or suspend it in terminal) | `C-z` |
| exit Emacs permanently | `C-x C-c` |

## Files

| | |
|---|---|
| **read** a file into Emacs | `C-x C-f` |
| **save** a file back to disk | `C-x C-s` |
| save **all** files | `C-x s` |
| **insert** contents of another file into this buffer | `C-x i` |
| replace this file with the file you really want | `C-x C-v` |
| write buffer to a specified file | `C-x C-w` |
| toggle read-only status of buffer | `C-x C-q` |

## Getting Help

The help system is simple. Type `C-h` (or `F1`) and follow the directions. If you are a first-time user, type `C-h t` for a **tutorial**.

| | |
|---|---|
| remove help window | `C-x 1` |
| scroll help window | `C-M-v` |
| apropos: show commands matching a string | `C-h a` |
| describe the function a key runs | `C-h k` |
| describe a function | `C-h f` |
| get mode-specific information | `C-h m` |

## Error Recovery

| | |
|---|---|
| **abort** partially typed or executing command | `C-g` |
| **recover** files lost by a system crash | `M-x recover-session` |
| **undo** an unwanted change | `C-x u`, `C-_` or `C-/` |
| restore a buffer to its original contents | `M-x revert-buffer` |
| redraw garbaged screen | `C-l` |

## Incremental Search

| | |
|---|---|
| search forward | `C-s` |
| search backward | `C-r` |
| regular expression search | `C-M-s` |
| reverse regular expression search | `C-M-r` |
| select previous search string | `M-p` |
| select next search string | `M-n` |
| exit incremental search | `RET` |
| undo effect of last character | `DEL` |
| abort current search | `C-g` |

Use `C-s` or `C-r` again to repeat the search in either direction. If Emacs is still searching, `C-g` cancels only the part not matched.

© 2025 Free Software Foundation, Inc. Permissions on back.

## Motion

| **entity to move over** | **backward** | **forward** |
|---|---|---|
| character | `C-b` | `C-f` |
| word | `M-b` | `M-f` |
| line | `C-p` | `C-n` |
| go to line beginning (or end) | `C-a` | `C-e` |
| sentence | `M-a` | `M-e` |
| paragraph | `M-{` | `M-}` |
| page | `C-x [` | `C-x ]` |
| sexp | `C-M-b` | `C-M-f` |
| function | `C-M-a` | `C-M-e` |
| go to buffer beginning (or end) | `M-<` | `M->` |

| | |
|---|---|
| scroll to next screen | `C-v` |
| scroll to previous screen | `M-v` |
| scroll left | `C-x <` |
| scroll right | `C-x >` |
| scroll current line to center, top, bottom | `C-l` |
| goto line | `M-g g` |
| goto char | `M-g c` |
| back to indentation | `M-m` |

## Killing and Deleting

| **entity to kill** | **backward** | **forward** |
|---|---|---|
| character (delete, not kill) | `DEL` | `C-d` |
| word | `M-DEL` | `M-d` |
| line (to end of) | `M-0 C-k` | `C-k` |
| sentence | `C-x DEL` | `M-k` |
| sexp | `M-- C-M-k` | `C-M-k` |

| | |
|---|---|
| kill **region** | `C-w` |
| copy region to kill ring | `M-w` |
| kill through next occurrence of *char* | `M-z` *char* |
| yank back last thing killed | `C-y` |
| replace last yank with previous kill | `M-y` |

## Marking

| | |
|---|---|
| set mark here | `C-@` or `C-SPC` |
| exchange point and mark | `C-x C-x` |
| set mark *arg* **words** away | `M-@` |
| mark **paragraph** | `M-h` |
| mark **page** | `C-x C-p` |
| mark **sexp** | `C-M-@` |
| mark **function** | `C-M-h` |
| mark entire **buffer** | `C-x h` |

## Query Replace

| | |
|---|---|
| interactively replace a text string | `M-%` |
| using regular expressions | `M-x query-replace-regexp` |

Valid responses in query-replace mode are

| | |
|---|---|
| **replace** this one, go on to next | `SPC` or `y` |
| replace this one, don't move | `,` |
| **skip** to next without replacing | `DEL` or `n` |
| replace all remaining matches | `!` |
| **back up** to the previous match | `^` |
| **exit** query-replace | `RET` |
| enter recursive edit (`C-M-c` to exit) | `C-r` |

## Multiple Windows

When two commands are shown, the second is a similar command for a frame instead of a window.

| | | |
|---|---|---|
| delete all other windows | `C-x 1` | `C-x 5 1` |
| split window, above and below | `C-x 2` | `C-x 5 2` |
| delete this window | `C-x 0` | `C-x 5 0` |
| split window, side by side | `C-x 3` | |
| scroll other window | `C-M-v` | |
| switch cursor to another window | `C-x o` | `C-x 5 o` |
| select buffer in other window | `C-x 4 b` | `C-x 5 b` |
| display buffer in other window | `C-x 4 C-o` | `C-x 5 C-o` |
| find file in other window | `C-x 4 f` | `C-x 5 f` |
| find file read-only in other window | `C-x 4 r` | `C-x 5 r` |
| run Dired in other window | `C-x 4 d` | `C-x 5 d` |
| find tag in other window | `C-x 4 .` | `C-x 5 .` |
| grow window taller | `C-x ^` | |
| shrink window narrower | `C-x {` | |
| grow window wider | `C-x }` | |

## Formatting

| | |
|---|---|
| indent current **line** (mode-dependent) | `TAB` |
| indent **region** (mode-dependent) | `C-M-\` |
| indent **sexp** (mode-dependent) | `C-M-q` |
| indent region rigidly *arg* columns | `C-x TAB` |
| indent for comment | `M-;` |
| insert newline after point | `C-o` |
| move rest of line vertically down | `C-M-o` |
| delete blank lines around point | `C-x C-o` |
| join line with previous (with arg, next) | `M-^` |
| delete all white space around point | `M-\` |
| put exactly one space at point | `M-SPC` |
| fill paragraph | `M-q` |
| set fill column to *arg* | `C-x f` |
| set prefix each line starts with | `C-x .` |

## Case Change

| | |
|---|---|
| uppercase word | `M-u` |
| lowercase word | `M-l` |
| capitalize word | `M-c` |
| uppercase region | `C-x C-u` |
| lowercase region | `C-x C-l` |

## The Minibuffer

The following keys are defined in the minibuffer.

| | |
|---|---|
| complete as much as possible | `TAB` |
| complete up to one word | `SPC` |
| complete and execute | `RET` |
| show possible completions | `?` |
| fetch previous minibuffer input | `M-p` |
| fetch later minibuffer input or default | `M-n` |
| regexp search backward through history | `M-r` |
| regexp search forward through history | `M-s` |
| abort command | `C-g` |

Type `C-x ESC ESC` to edit and repeat the last command that used the minibuffer. Type `F10` to activate menu bar items on text terminals.

# GNU Emacs Reference Card

## Buffers

| | |
|---|---|
| select another buffer | `C-x b` |
| list all buffers | `C-x C-b` |
| kill a buffer | `C-x k` |

## Transposing

| | |
|---|---|
| transpose **characters** | `C-t` |
| transpose **words** | `M-t` |
| transpose **lines** | `C-x C-t` |
| transpose **sexps** | `C-M-t` |

## Spelling Check

| | |
|---|---|
| check spelling of current word | `M-$` |
| check spelling of all words in region | `M-x ispell-region` |
| check spelling of entire buffer | `M-x ispell-buffer` |
| toggle on-the-fly spell checking | `M-x flyspell-mode` |

## Tags

| | |
|---|---|
| find a tag (a definition) | `M-.` |
| specify a new tags file | `M-x visit-tags-table` |
| regexp search on all files in tags table | `M-x tags-search` |
| run query-replace on all the files | `M-x tags-query-replace` |

## Shells

| | |
|---|---|
| execute a shell command | `M-!` |
| execute a shell command asynchronously | `M-&` |
| run a shell command on the region | `M-\|` |
| filter region through a shell command | `C-u M-\|` |
| start a shell in window `*shell*` | `M-x shell` |

## Rectangles

| | |
|---|---|
| copy rectangle to register | `C-x r r` |
| kill rectangle | `C-x r k` |
| yank rectangle | `C-x r y` |
| open rectangle, shifting text right | `C-x r o` |
| blank out rectangle | `C-x r c` |
| prefix each line with a string | `C-x r t` |

## Abbrevs

| | |
|---|---|
| add global abbrev | `C-x a g` |
| add mode-local abbrev | `C-x a l` |
| add global expansion for this abbrev | `C-x a i g` |
| add mode-local expansion for this abbrev | `C-x a i l` |
| explicitly expand abbrev | `C-x a e` |
| expand previous word dynamically | `M-/` |

## Miscellaneous

| | |
|---|---|
| numeric argument | `C-u` *num* |
| negative argument | `M--` |
| quoted insert | `C-q` *char* |

## Regular Expressions

| | |
|---|---|
| any single character except a newline | `.` (dot) |
| zero or more repeats | `*` |
| one or more repeats | `+` |
| zero or one repeat | `?` |
| quote special characters | `\` |
| quote regular expression special character *c* | `\`*c* |
| alternative (“or”) | `\|` |
| grouping | `\( ... \)` |
| shy grouping | `\(?: ... \)` |
| explicit numbered grouping | `\(?NUM: ... \)` |
| same text as *n*th group | `\`*n* |
| at word break | `\b` |
| not at word break | `\B` |

| **entity** | **match start** | **match end** |
|---|---|---|
| line | `^` | `$` |
| word | `\<` | `\>` |
| symbol | `\_<` | `\_>` |
| buffer | `\‘` | `\’` |
| **class of characters** | **match these** | **match others** |
| explicit set | `[ ... ]` | `[^ ... ]` |
| word-syntax character | `\w` | `\W` |
| character with syntax *c* | `\s`*c* | `\S`*c* |
| character with category *c* | `\c`*c* | `\C`*c* |

## International Character Sets

| | |
|---|---|
| specify principal language | `C-x RET l` |
| show all input methods | `M-x list-input-methods` |
| enable or disable input method | `C-\` |
| set coding system for next command | `C-x RET c` |
| show all coding systems | `M-x list-coding-systems` |
| choose preferred coding system | `M-x prefer-coding-system` |

## Info

| | |
|---|---|
| enter the Info documentation reader | `C-h i` |
| find specified function or variable in Info | `C-h S` |

Moving within a node:

| | |
|---|---|
| scroll forward | `SPC` |
| scroll reverse | `DEL` |
| beginning of node | `b` |

Moving between nodes:

| | |
|---|---|
| **next** node | `n` |
| **previous** node | `p` |
| move **up** | `u` |
| select menu item by name | `m` |
| select *n*th menu item by number (1–9) | *n* |
| follow cross reference (return with `l`) | `f` |
| return to last node you saw | `l` |
| return to directory node | `d` |
| go to top node of Info file | `t` |
| go to any node by name | `g` |

Other:

| | |
|---|---|
| run Info **tutorial** | `h` |
| look up a subject in the indices | `i` |
| search nodes for regexp | `s` |
| **quit** Info | `q` |

## Registers

| | |
|---|---|
| save region in register | `C-x r s` |
| insert register contents into buffer | `C-x r i` |
| save value of point in register | `C-x r SPC` |
| jump to point saved in register | `C-x r j` |

## Keyboard Macros

| | |
|---|---|
| **start** defining a keyboard macro | `C-x (` |
| **end** keyboard macro definition | `C-x )` |
| **execute** last-defined keyboard macro | `C-x e` |
| append to last keyboard macro | `C-u C-x (` |
| name last keyboard macro | `M-x name-last-kbd-macro` |
| insert Lisp definition in buffer | `M-x insert-kbd-macro` |

## Commands Dealing with Emacs Lisp

| | |
|---|---|
| eval **sexp** before point | `C-x C-e` |
| eval current **defun** | `C-M-x` |
| eval **region** | `M-x eval-region` |
| read and eval minibuffer | `M-:` |
| load a Lisp library from **load-path** | `M-x load-library` |

## Simple Customization

| | |
|---|---|
| customize variables and faces | `M-x customize` |

Making global key bindings in Emacs Lisp (example):

```
(global-set-key (kbd "C-c g") 'search-forward)
(global-set-key (kbd "M-#") 'query-replace-regexp)
```

## Writing Commands

`(defun` *command-name* `(`*args*`)`
`"`*documentation*`"` `(interactive "`*template*`")`
*body*`)`

An example:

```
(defun this-line-to-top-of-window (line)
  "Reposition current line to top of window.
With prefix argument LINE, put point on LINE."
  (interactive "P")
  (recenter (if (null line)
                0
              (prefix-numeric-value line))))
```

The `interactive` spec says how to read arguments interactively. Type `C-h f interactive RET` for more details.

Copyright © 2025 Free Software Foundation, Inc.
For GNU Emacs version 30
Designed by Stephen Gildea

Released under the terms of the GNU General Public License version 3 or later.

For more Emacs documentation, and the TEX source for this card, see the Emacs distribution, or `https://www.gnu.org/software/emacs`